

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ
ФЕДЕРАЦИИ**
**Федеральное государственное автономное
образовательное учреждение высшего образования
«СЕВЕРОКАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

Кафедра инфокоммуникаций

Институт цифрового развития

ОТЧЁТ

по лабораторной работе №2.4

Дисциплина: «Основы кроссплатформенного программирования»

Тема: «Работа со списками в языке Python»

Выполнил: студент 1 курса

группы ИВТ-б-о-21-1

Горшков Виталий Игоревич

Ставрополь 2022

Выполнение работы:

1. Создал репозиторий в GitHub «rep 2.5» в который добавил .gitignore, который дополнил правила для работы с IDE PyCharm с ЯП Python, выбрал лицензию MIT, клонировал его на лок. сервер и организовал в соответствии с моделью ветвления git-flow.

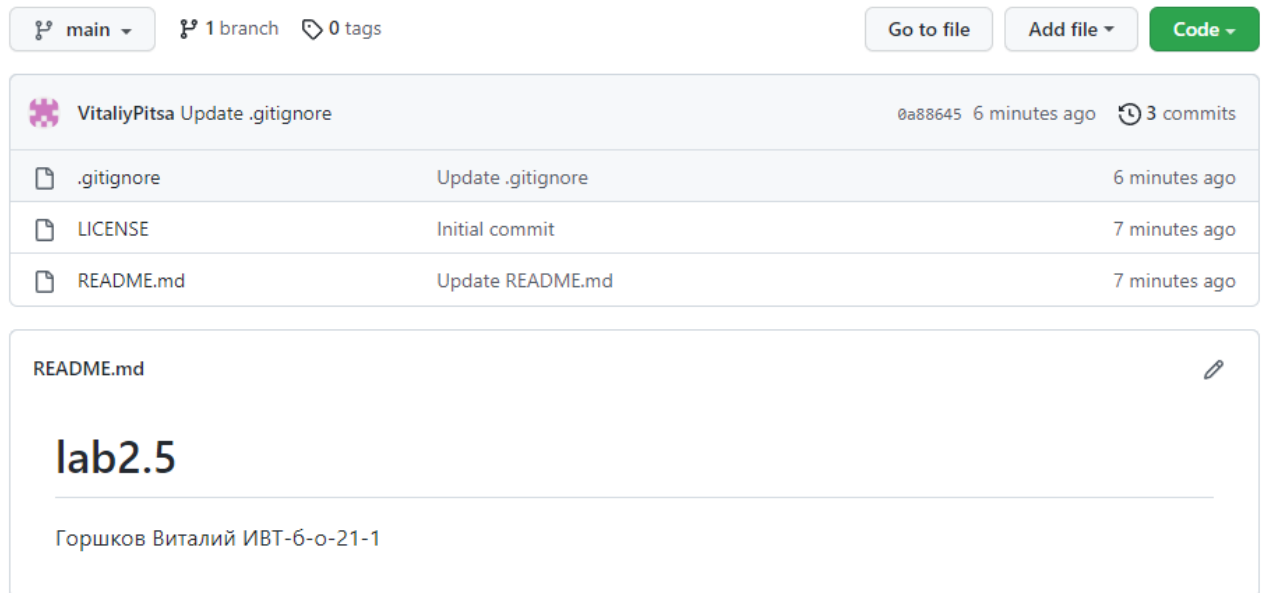


Рисунок 1 Создание репозитория

```
C:\lab2.5>git clone https://github.com/VitaliyPitsa/lab2.5.git
Cloning into 'lab2.5'...
remote: Enumerating objects: 11, done.
remote: Counting objects: 100% (11/11), done.
remote: Compressing objects: 100% (10/10), done.
remote: Total 11 (delta 2), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (11/11), 5.09 KiB | 326.00 KiB/s, done.
Resolving deltas: 100% (2/2), done.
```

Рисунок 2 Клонирование репозитория

```
C:\lab2.5\lab2.5>git flow init

Which branch should be used for bringing forth production releases?
- main
Branch name for production releases: [main]
Branch name for "next release" development: [develop]

How to name your supporting branch prefixes?
Feature branches? [feature/]
Bugfix branches? [bugfix/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? []
Hooks and filters directory? [C:/lab2.5/lab2.5/.git/hooks]
```

Рисунок 3 Организация репозитория в соответствии с моделью ветвления
git-flow

```
1 .idea/
2 # Created by https://www.toptal.com/developers/gitignore/api/python,pycharm
3 # Edit at https://www.toptal.com/developers/gitignore?templates=python,pycharm
4
5 ### PyCharm ###
6 # Covers JetBrains IDEs: IntelliJ, RubyMine, PhpStorm, AppCode, PyCharm, CLion, Android Studio, WebStorm and Rider
7 # Reference: https://intellij-support.jetbrains.com/hc/en-us/articles/206544839
8
9 # User-specific stuff
10 .idea/**/workspace.xml
11 .idea/**/tasks.xml
12 .idea/**/usage.statistics.xml
13 .idea/**/dictionaries
14 .idea/**/shelf
15
16 # AWS User-specific
17 .idea/**/aws.xml
18
19 # Generated files
20 .idea/**/contentModel.xml
21
22 # Sensitive or high-churn files
23 .idea/**/dataSources/
24 .idea/**/dataSources.ids
25 .idea/**/dataSources.local.xml
26 .idea/**/sqlDataSources.xml
27 .idea/**/dynamic.xml
28 .idea/**/uiDesigner.xml
29 .idea/**/dbnavigator.xml
30
31 # Gradle
32 .idea/**/gradle.xml
33 .idea/**/libraries
34
35 # Gradle and Maven with auto-import
36 # When using Gradle or Maven with auto-import, you should exclude module files,
37 # since they will be recreated, and may cause churn. Uncomment if using
38 # auto-import.
39 # .idea/artifacts
40 # .idea/compiler.xml
```

Рисунок 4 Изменение .gitignore

2. Создал проект PyCharm в папке репозитория, проработал примеры ЛР.

2 3 1 5 6 4 8 7 9 2
12

Рисунок 5 Рез-т выполнения программы

3. (8 вариант). Выполнил индивидуальное задание.

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import sys

if __name__ == '__main__':
    arr = tuple(map(int, input().split()))
    print("кортеж:", arr)

    for i in range(len(arr) - 2):
        e1 = arr[i]
        e2 = arr[i + 1]
        e3 = arr[i + 2]
        if e1 < e2 > e3:
            print("индексы элементов кортежа:", i, i + 1, i + 2, " их значения:", e1, e2, e3)
            break
```

```
1 4 1 2 4 5
кортеж: (1, 4, 1, 2, 4, 5)
индексы элементов кортежа: 0 1 2   их значения: 1 4 1

Process finished with exit code 0
|
```

Рисунок 5 Вывод программы индивидуального задания

4. Сделал коммит, выполнил слияние с веткой main, и запустил изменения в уд. репозиторий.

```
C:\lab2.5\lab2.5>git add .

C:\lab2.5\lab2.5>git commit -m "1commit"
[develop b02d15f] 1commit
 2 files changed, 36 insertions(+)
 create mode 100644 individual1.py
 create mode 100644 primer1.py

C:\lab2.5\lab2.5>git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
```

Рисунок 6 коммит изменений и переход на ветку main

```
C:\lab2.5\lab2.5>git merge develop
Updating 0a88645..b02d15f
Fast-forward
 individual1.py | 16 +++++
 primer1.py      | 20 +++++
 2 files changed, 36 insertions(+)
 create mode 100644 individual1.py
 create mode 100644 primer1.py
```

Рисунок 7 Слияние ветки main с develop

```

C:\lab2.5\lab2.5>git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 1015 bytes | 1015.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/VitaliyPitsa/lab2.5.git
   0a88645..b02d15f  main -> main

```

Рисунок 8 Пуш изменений на удаленный сервер






| | | |
|--|-------------------|----------------|
|  .gitignore | Update .gitignore | 27 minutes ago |
|  LICENSE | Initial commit | 29 minutes ago |
|  README.md | Update README.md | 28 minutes ago |
|  individual1.py | 1commit | 1 minute ago |
|  primer1.py | 1commit | 1 minute ago |

Рисунок 9 Изменения на удаленном сервере

Контр. вопросы и ответы на них:

1. Что такое кортежи в языке Python?

Кортеж (tuple) – это неизменяемая структура данных, которая по своему подобию очень похожа на список.

2. Каково назначение кортежей в языке Python?

Существует несколько причин, по которым стоит использовать кортежи вместо списков. Одна из них – это обезопасить данные от случайного изменения. Если мы получили откуда-то массив данных, и у нас есть желание поработать с ним, но при этом непосредственно менять данные мы не собираемся, тогда, это как раз тот случай, когда кортежи придутся как нельзя кстати. Кортежи в памяти занимают меньший объем по сравнению со списками. Кортежи работают быстрее, чем списки

3. Как осуществляется создание кортежей?

a = ()

b = tuple()

4. Как осуществляется доступ к элементам кортежа?

Доступ к элементам кортежа осуществляется также как к элементам списка – через указание индекса.

5. Зачем нужна распаковка (деструктуризация) кортежа?

Обращение по индексу, это не самый удобный способ работы с кортежами. Дело в том, что кортежи часто содержат значения разных типов, и помнить, по какому индексу что лежит — очень непросто.

6. Какую роль играют кортежи в множественном присваивании?

Используя множественное присваивание, можно проверить интересный трюк: обмен значениями между двумя переменными.

7. Как выбрать элементы кортежа с помощью среза?

С помощью операции взятия среза можно получить другой кортеж. Общая форма операции взятия среза для кортежа следующая

$T2 = T1[i:j]$

здесь

- $T2$ – новый кортеж, который получается из кортежа $T1$;
- $T1$ – исходный кортеж, для которого происходит срез;
- i, j – соответственно нижняя и верхняя границы среза. Фактически

берутся ко вниманию элементы, лежащие на позициях $i, i+1, \dots, j-1$. Значение j определяет позицию за последним элементом среза.

8. Как выполняется конкатенация и повторение кортежей?

Для кортежей можно выполнять операцию конкатенации, которая обозначается символом $+$.

$T3 = T1 + T2$

9. Как выполняется обход элементов кортежа?

Элементы кортежа можно последовательно просмотреть с помощью операторов цикла `while` или `for`.

10. Как проверить принадлежность элемента кортежу?

Проверка вхождения элемента в кортеж - оператор `in`.

11. Какие методы работы с кортежами Вам известны?

`index()`, `count()`.

12. Допустимо ли использование функций агрегации таких как `len()`, `sum()` и т. д. при работе с кортежами?

Доступно.

13. Как создать кортеж с помощью спискового включения.

Так же как и список.